

Exact Decoding on Latent Variable Conditional Models is NP-Hard

Xu Sun

Abstract—Latent variable conditional models, including the latent conditional random fields as a special case, are popular models for many natural language processing and vision processing tasks. The computational complexity of the exact decoding/inference in latent conditional random fields is unclear. In this paper, we try to clarify the computational complexity of the exact decoding. We analyze the complexity and demonstrate that it is an NP-hard problem even on a sequential labeling setting. Furthermore, we propose the latent-dynamic inference (LDI-Naive) method and its bounded version (LDI-Bounded), which are able to perform exact-inference or almost-exact-inference by using top- n search and dynamic programming.

Index Terms—Latent Variable Conditional Models, Computational Complexity Analysis, Exact Decoding.

I. INTRODUCTION

Real-world problems may contain hidden structures that are difficult to be captured by conventional structured classification models without latent variables. For example, in the syntactic parsing task for natural language, the hidden structures can be refined grammars that are unobservable in the supervised training data [1]. In the gesture recognition task of the computational vision area, there are also hidden structures which are crucial for successful gesture recognition [2]. There are also plenty of hidden structure examples in other tasks among different areas [3], [4], [5], [6], [7], [8], [9].

In such cases, models that exploit latent variables are advantageous in learning [3], [4], [1], [5], [6], [10], [11], [12], [9]. As a representative structured classification model with latent variables, the latent conditional random fields (LCRFs) have become widely-used for performing a variety of tasks with hidden structures, e.g., vision recognition [4], and syntactic parsing [1], [9]. For example, [4] demonstrated that LCRF models can learn latent structures of vision recognition problems efficiently, and outperform several widely-used conventional models, such as support vector machines (SVMs), conditional random fields (CRFs) and hidden Markov models (HMMs). [1] and [9] reported on a syntactic parsing task that LCRF models can learn more accurate grammars than models that use conventional techniques without latent variables.

Exact inference in the latent conditional models is a remaining problem. In conventional models, such as conditional random fields (CRFs), the optimal labeling can be efficiently obtained by dynamic programming algorithms (for example, the Viterbi algorithm). Nevertheless, for latent conditional random fields, the inference is not straightforward, because of the inclusion of latent variables. The computational complexity

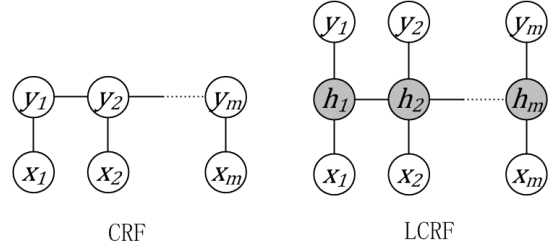


Fig. 1. Comparison between CRF models and LCRF models on the training stage. x represents the observation sequence, y represents labels and h represents the latent variables assigned to the labels. Note that only the white circles are observed variables. Also, only the links with the current observations are depicted, but for both models, long range dependencies are possible.

of inference in LCRFs, with a disjoint association among latent variables and a linear chain structure, is still unclear. In this paper, we will show that such kind of inference is actually NP-hard. This is a critical limitation on the real-world applications of LCRFs. Most of the previous applications of LCRFs tried to make simplified approximations on the inference [13], [4], [1], but the inference accuracy can be limited.

Although we will show that the inference in LCRFs is NP-hard, in real-world applications, we have an interesting observation on LCRF models: they normally have a highly concentrated probability distribution. The major probability is distributed on top- n ranked latent-labelings. In this paper, we try to make systematic analysis on this probability concentration phenomenon. We will show that probability concentration is reasonable and expected in latent conditional random fields. Based on this analysis, we will propose a new inference algorithm: latent dynamic inference (LDI), by systematically combining efficient top- n search with dynamic programming. The LDI is an exact inference method, producing the most probable label sequence. In addition, for speeding up the inference, we will also propose a bounded version of the LDI algorithm.

II. LATENT CONDITIONAL RANDOM FIELDS

Given the training data, the task is to learn a mapping between a sequence of observations $\mathbf{x} = x_1, x_2, \dots, x_m$ and a sequence of labels $\mathbf{y} = y_1, y_2, \dots, y_m$. Each y_j is a class label for the j 'th token of a word sequence, and is a member of a set \mathcal{Y} of possible class labels. For each sequence, the model also assumes a sequence of latent variables $\mathbf{h} = h_1, h_2, \dots, h_m$, which is unobservable in training examples. See Figure 1 for the comparison between CRFs and LCRFs.

X. Sun is with Department of Computer Science, School of EECS, Peking University. E-mail: xusun@pku.edu.cn

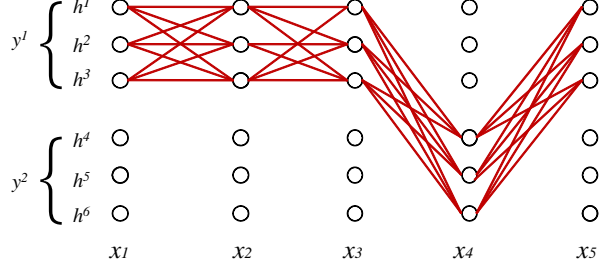


Fig. 2. In this example, the probability of the labeling “ y^1, y^1, y^1, y^2, y^1 ” is summed over all of its latent-labelings (represented by the red paths from position x_1 to x_5). The label y^1 has latent variables h^1 to h^3 . The label y^2 has latent variables h^4 to h^6 .

The LCRF model is defined as follows [4]:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \triangleq \sum_{\mathbf{h}} P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \mathbf{w}) P(\mathbf{h}|\mathbf{x}, \mathbf{w}),$$

where \mathbf{w} represents the parameter vector of the model. To make the training efficient, a restriction is made for the model: for each label, the latent variables associated with it have no intersection with the latent variables from other labels [4], [1]. This simplification is also a popular practice in other latent conditional models, including hidden-state conditional random fields (HCRF) [14]. Each h is a member in a set $\mathcal{H}(y)$ of possible latent variables for the class label y , and $\mathcal{H}(y^j) \cap \mathcal{H}(y^k) = \emptyset$ if $y^j \neq y^k$. \mathcal{H} is defined as the set of all possible latent variables; i.e., the union of all $\mathcal{H}(y)$ sets: $\mathcal{H} = \cup_{y \in \mathcal{Y}} \mathcal{H}(y)$. In other words, h can have any value from \mathcal{H} , but $P(y|h)$ is zero except for only one of y in \mathcal{Y} . The disjoint restriction indicates a discrete simplification of $P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \mathbf{w})$:

$$\begin{aligned} P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \mathbf{w}) &= 1 \quad \text{if } \mathbf{h} \in \mathcal{H}(y_1) \times \dots \times \mathcal{H}(y_m) \\ P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \mathbf{w}) &= 0 \quad \text{if } \mathbf{h} \notin \mathcal{H}(y_1) \times \dots \times \mathcal{H}(y_m) \end{aligned}$$

where m is the length of the labeling¹: $m = |\mathbf{y}|$. The formula $\mathbf{h} \in \mathcal{H}(y_1) \times \dots \times \mathcal{H}(y_m)$ indicates that the latent-labeling \mathbf{h} is a latent-labeling of the labeling \mathbf{y} , which can be more formally defined as follows:

$$\mathbf{h} \in \mathcal{H}(y_1) \times \dots \times \mathcal{H}(y_m) \iff h_j \in \mathcal{H}(y_j) \text{ for } j = 1, \dots, m$$

Since sequences that have any $h_j \notin \mathcal{H}(y_j)$ will by definition have $P(\mathbf{y}|\mathbf{h}_j, \mathbf{x}, \mathbf{w}) = 0$, the model can be simplified as:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \triangleq \sum_{\mathbf{h} \in \mathcal{H}(y_1) \times \dots \times \mathcal{H}(y_m)} P(\mathbf{h}|\mathbf{x}, \mathbf{w}). \quad (1)$$

In Figure 2, an example is shown to illustrate the way to compute $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$ for a given labeling by summing over probabilities of all its latent-labelings. The item $P(\mathbf{h}|\mathbf{x}, \mathbf{w})$ is defined by the usual conditional random field formulation:

$$P(\mathbf{h}|\mathbf{x}, \mathbf{w}) = \frac{\exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x})\}}{\sum_{\mathbf{h}' \in \mathcal{H} \times \dots \times \mathcal{H}} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{h}', \mathbf{x})\}}, \quad (2)$$

in which $\mathbf{f}(\mathbf{h}, \mathbf{x})$ is a global feature vector. LCRF models can be seen as a natural extension of CRF models, and CRF

¹A labeling is a sequence of predicted classes: $\mathbf{y} = y_1, y_2, \dots, y_m$.

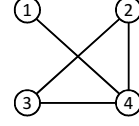


Fig. 3. An example of an undirected graph that contains one maximal clique. This example will be used for complexity analysis.

models can be seen as a special case of LCRFs that employ only one latent variable for each label (i.e., $|\mathcal{H}(y)| = 1$ for each y in \mathcal{Y}). The global feature vector can be calculated by summing over all its local feature vectors:

$$\mathbf{f}(\mathbf{h}, \mathbf{x}) = \sum_{i=1, \dots, m} \mathbf{f}_i(\mathbf{h}, \mathbf{x}) + \sum_{i=1, \dots, m-1} \mathbf{f}_{i,i+1}(\mathbf{h}, \mathbf{x}), \quad (3)$$

in which $\mathbf{f}_i(\mathbf{h}, \mathbf{x})$ represents a local feature vector that depends only on h_i and \mathbf{x} . The $\mathbf{f}_{i,i+1}(\mathbf{h}, \mathbf{x})$ represents the local feature vector that depends only on h_i , h_{i+1} , and \mathbf{x} .

Given a training set consisting of n labeled sequences, $(\mathbf{x}_i, \mathbf{y}_i)$, for $i = 1 \dots n$, parameter estimation is performed by optimizing the objective function,

$$L(\mathbf{w}) = \sum_{i=1}^n \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) - R(\mathbf{w}).$$

The first term of this equation represents a conditional marginal log-likelihood of a training data. The second term is a regularizer that is used for reducing overfitting in parameter estimation. L_2 regularization is often used, and it is defined as: $R(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2\sigma^2}$, where σ is the hyper-parameter that controls the degree of regularization.

III. COMPLEXITY OF EXACT INFERENCE

In this section, we will make a formal analysis of the computational complexity of inference in LCRFs. The computational complexity analysis will be based on a reduction from a well-known NP-hard problem to the inference problem on LCRFs. We assume that the reader has basic background in the complexity theory, including the notions of *NP* and *NP-hardness* [15], [16], [17], [18]. Normally, a minimum requirement for an algorithm to be considered as *efficient* is that its running time is *polynomial*: $O(n^c)$, where c is a constant real value and n is the size of the input. It is believed that the NP-hard problems cannot be solved in polynomial time, although there has been no proof of a super-polynomial lower bound.

The well-known **Maximum-Clique** problem is an NP-hard problem. Figure 3 gives an example of the maximum clique problem. The graph consists of 4 connected nodes, and the number of maximum-clique nodes is 3, because the maximum clique is $\{2, 3, 4\}$.

We will prove that the exact inference in LCRFs is an NP-hard problem, and hence, finding the labeling with the maximum probability on LCRFs is likely to be intractable. Inspired by the consensus string problem on hidden Markov models [19], we establish the hardness analysis of the problem by a reduction from the maximum clique problem.

Definition 1. Maximum clique problem

Instance: An undirected graph $G = \{\mathcal{V}, \mathcal{D}\}$ with the indexed nodes $\mathcal{V} = \{1, 2, \dots, |\mathcal{V}|\}$ and the edge set \mathcal{D} defined among \mathcal{V} .

Question: What is the size of the maximum clique of G ?

Since the \mathbf{x} and \mathbf{w} are fixed for the inference task, we can simplify the denotation and define *node scores* $\varphi_i(\mathbf{h}) = \exp\{\mathbf{w}^\top \mathbf{f}_i(\mathbf{h}, \mathbf{x})\}$ and *edge scores* $\varphi_{i,i+1}(\mathbf{h}) = \exp\{\mathbf{w}^\top \mathbf{f}_{i,i+1}(\mathbf{h}, \mathbf{x})\}$. Then, based on Eq. 3, we can reformulate Eq. 2 as follows:

$$P(\mathbf{h}|\mathbf{x}, \mathbf{w}) = \frac{\prod_{i=1, \dots, m} \varphi_i(\mathbf{h}) \cdot \prod_{i=1, \dots, m-1} \varphi_{i,i+1}(\mathbf{h})}{\sum_{\mathbf{h}' \in \mathcal{H} \times \dots \times \mathcal{H}} \left\{ \prod_{i=1, \dots, m} \varphi_i(\mathbf{h}') \cdot \prod_{i=1, \dots, m-1} \varphi_{i,i+1}(\mathbf{h}') \right\}}.$$

This reformulation indicates that the probability of a latent labeling \mathbf{h} can be computed by the *path-score* of \mathbf{h} , and divided by the summation of all path-scores. The *path-score* of \mathbf{h} is defined by the multiplication over all the node scores and edges scores of \mathbf{h} .

Definition 2. Inference in LCRFs

Instance: A latent variable lattice (e.g., see Figure 2) $M = \{\mathbf{x}, \mathcal{Y}, \mathcal{H}, \Phi_n, \Phi_e\}$: For input sequence \mathbf{x} , $|\mathbf{x}| = m$; \mathcal{Y} and \mathcal{H} are defined as before; Φ_n is a $m \times |\mathcal{H}|$ matrix, in which an element $\Phi_n(i, j)$ represents a real-value node score φ_n for the latent variable $j \in \mathcal{H}$ on the position i (corresponding to x_i); Φ_e is a $m \times |\mathcal{H}| \times |\mathcal{H}|$ three-dimensional array, in which an element $\Phi_e(i, j, k)$ represents a real-value edge score φ_e for the edge (transition) between the latent variables $j \in \mathcal{H}$ and $k \in \mathcal{H}$, which are on the positions i and $i + 1$, respectively.

Question: With the $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$ being defined in Eq. 1, what is the optimal labeling \mathbf{y}^* in M such that $\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, \mathbf{w})$?

Based on those definitions, we have the following theorem:

Theorem 1. The computational complexity of the exact inference in LCRFs, $\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, \mathbf{w})$, is **NP-hard**.

This means the exact inference on LCRFs is an NP-hard problem. The proof is extended from related work on complexity analysis [20], [21]. In [20], the complexity analysis is based on Bayesian networks. In [21], the analysis is based on grammar models. We extend the complexity analysis to LCRFs.

A. Proof

Here, we prove the Theorem 1. For an undirected graph $G = \{\mathcal{V}, \mathcal{D}\}$, we present the construction of a corresponding latent conditional model M_G from G , so that the size of the maximum clique of G is proportional to the probability of the optimal labeling of M_G . We set the length of the input sequence: $m = |\mathcal{V}|$. We set $\mathcal{Y} = \{E, N\}$. We set $\mathcal{H} = \{E^1, E^2, \dots, E^{|\mathcal{V}|}, N^1, N^2, \dots, N^{|\mathcal{V}|}\}$, and the disjoint association between \mathcal{Y} and \mathcal{H} is as follows: $\mathcal{H}(E) = \{E^1, E^2, \dots, E^{|\mathcal{V}|}\}$, and $\mathcal{H}(N) = \{N^1, N^2, \dots, N^{|\mathcal{V}|}\}$. The

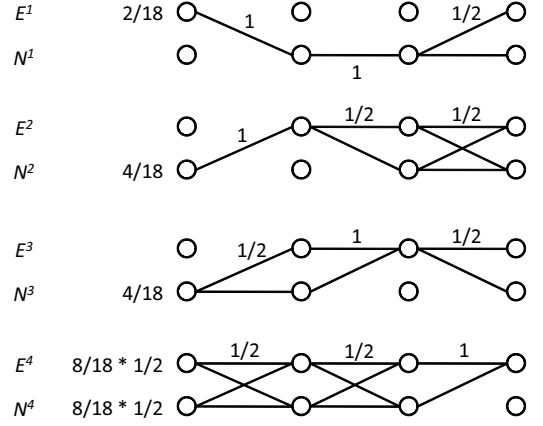


Fig. 4. The latent conditional model M_G constructed from the graph G in Fig 3.

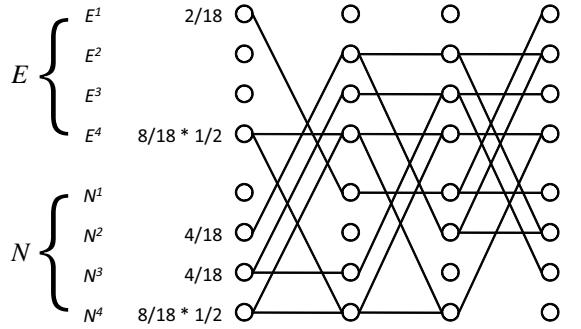


Fig. 5. The same latent conditional model M_G transformed from Fig 4. This figure is to show that the latent conditional model constructed in Fig 4 is valid.

settings on node-scores and edge-scores on the lattice are as follows: We group paths with non-zero probability in the lattice into $|\mathcal{V}|$ layers, such that each layer contains two horizontal rows of the lattice, and there is a layer corresponding to a node in \mathcal{V} . In what follows, we say a path is valid if it has a non-zero probability, and a node is valid if there is at least one valid path that passes the node. The properties of a layer are as follows:

There are a total of $|\mathcal{V}|$ layers $L = \{L_1, L_2, \dots, L_{|\mathcal{V}|}\}$. A layer L_i corresponds to a node $i \in \mathcal{V}$. Any paths that go through nodes from different layers are not valid. The layer L_i contains only E^i and N^i . All *paths* (latent-labelings) in the layer should pass the E^i and *avoid* the N^i on the i 'th position: $\Phi_n(i, E^i) = 1$ and $\Phi_n(i, N^i) = 0$. For any position k other than i in the layer L_i , if the node k in G is connected to node i in G , $(k, i) \in \mathcal{D}$, then both the E^i and N^i are valid on the position k : $\Phi_n(k, E^i) = 1$ and $\Phi_n(k, N^i) = 1$. Otherwise, only the N^i is valid. For the edge scores Φ_e , all the edges involving an invalid node (with node score of 0) will become an invalid edge (with edge score of 0). For any of the remaining valid edges (j, k) , its edge score is 1/2 if node j starts two valid edges and 1 if it starts only one valid edge. Finally, we adjust the node scores of the beginning nodes. If

both of the beginning nodes are valid in a layer, both of the nodes will have the probability of $1/2$. The node scores of the beginning nodes of the layer L_i are multiplied a factor $\delta(i)$ that is related to the degree of the node i in the graph G : $\delta_i = \frac{2^{\deg(i)}}{\sum_{v \in \mathcal{V}} 2^{\deg(v)}}$.

Given any valid path \mathbf{h} in the constructed model, it can be proved that the total number of valid paths are α , with $\alpha = \sum_{v \in \mathcal{V}} 2^{\deg(v)}$. Also, all the valid paths have the same path-score, $\frac{1}{\alpha}$. The summation of all the path scores is exactly 1. An example for constructing a latent conditional model based on the previous simple graph (see Figure 3) is shown in Figure 4. In the figure, we only show the valid edges for simplicity. The path-score of any valid path in Figure 4 is $1/18$. The model structure in Figure 4 is a valid type of structure of LCRFs (see Figure 5). The reduction is always possible in polynomial time.

Lemma 2. *If a labeling \mathbf{y} in M_G has a probability of c/α (c is a integer with $c \geq 1$), then the graph G must have a maximum clique with the size of at least c .*

Since each valid latent-labeling has the identical probability of $1/\alpha$, $P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = c/\alpha$ means that \mathbf{y} must have c different latent-labelings. A clear property of M_G is that one layer can only produce, at most, one latent-labeling for a specified \mathbf{y} . Therefore, each of the c latent-labelings of \mathbf{y} must come from c different layers. Suppose that the c different layers are $L_{x^1}, L_{x^2}, \dots, L_{x^c}$, and $\mathcal{X} = \{x^1, \dots, x^c\}$ is the set of indexes of the c different layers. If \mathbf{y} contains a latent-labeling from a layer L_i , then E^i must be chosen on the i 'th position. Therefore, \mathbf{y} must have the label E on at least c different positions x^1, \dots, x^c . It indicates that each of the c latent-labelings of \mathbf{y} must have E^i on at least c different positions x^1, \dots, x^c , and for each case, the node i in G is connected to all the $c-1$ nodes indexed by the set $\mathcal{X} - \{i\}$. Thus, the nodes x^1, \dots, x^c in G are connected to each other, and they form a clique of the size c in G .

Lemma 3. *If G has a clique of the size c , there must be a labeling in M_G with the probability of at least c/α .*

Suppose that the c nodes of the clique in G are indexed by a set $\mathcal{X} = \{x^1, \dots, x^c\}$; then, in each layer L_i with $i \in \mathcal{X}$, there must be a valid \mathbf{h}_i that passes the E^i on all of the positions of \mathcal{X} . The N^i is always valid for all positions, except the position i . Especially, N^i is valid for any position $k \notin \mathcal{X}$. On the other hand, E^i is valid at each position $k' \in \mathcal{X}$ since \mathcal{X} forms a clique in G . Hence, for L_i , the \mathbf{h}_i described as follows must be valid: \mathbf{h}_i passes E^i for all positions in \mathcal{X} and passes N^i for all positions not in \mathcal{X} (i.e., $\mathcal{V} - \mathcal{X}$). The c latent-labelings from different layers are consistent and belong to an identical labeling, \mathbf{y} . Since each latent-labeling has the probability of $1/\alpha$, the \mathbf{y} has the probability of at least c/α .

Combining the two lemmas, we can see that the graph G has a maximum clique of the size c if and only if the model M_G has a maximum-probability labeling with the probability c/α . Since the reduction is always possible in polynomial time, we have Theorem 1.

IV. A PRACTICAL SOLUTION BASED ON PROBABILITY CONCENTRATION

We have shown that exact inference in latent conditional models is an NP-hard problem. Nevertheless, we try to solve this difficult problem based on an interesting observation. In real world applications, we have an observation on LCRFs: They normally have a highly concentrated probability distribution. That is, most of the probability mass is distributed on top- n ranked latent labelings.

A. Probability Concentration from Optimization

To formally analyze the reason of the probability concentration on LCRFs, we first analyze the optimization process on LCRFs. Since the optimization process on LCRFs is based on the gradient information of its objective function, it is critical to analyze the trends of the gradient formulation of the LCRF objective function. In training LCRFs, people perform gradient ascent for maximizing the objective function. The log-likelihood portion of the objective function is as follows:

$$\begin{aligned} \mathcal{L} &= \log \left\{ \frac{\sum_{\mathbf{h} \in \mathbf{y}^*} \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x})]}{\sum_{\mathbf{h}' \in \mathbf{y}^*} \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{h}', \mathbf{x})]} \right\} \\ &= \log \left\{ \sum_{\mathbf{h} \in \mathbf{y}^*} \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x})] \right\} - \log \left\{ \sum_{\mathbf{h}' \in \mathbf{y}^*} \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{h}', \mathbf{x})] \right\}. \end{aligned}$$

Hence, its gradient is as follows:

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L} &= \sum_{\mathbf{h} \in \mathbf{y}^*} \{P^*(\mathbf{h}) \mathbf{f}(\mathbf{h}, \mathbf{x})\} - \sum_{\mathbf{h}' \in \mathbf{y}^*} \{P(\mathbf{h}') \mathbf{f}(\mathbf{h}', \mathbf{x})\} \\ &= \sum_{\mathbf{h} \in \mathbf{y}^*} \{[P^*(\mathbf{h}) - P(\mathbf{h})] \mathbf{f}(\mathbf{h}, \mathbf{x})\} - \sum_{\mathbf{h}' \notin \mathbf{y}^*} \{P(\mathbf{h}') \mathbf{f}(\mathbf{h}', \mathbf{x})\}, \end{aligned} \quad (4)$$

where $P^*(\mathbf{h})$ is the probability of \mathbf{h} with regard to only \mathbf{y}^* . In other words,

$$P^*(\mathbf{h}) = \frac{\exp[\mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x})]}{\sum_{\mathbf{h} \in \mathbf{y}^*} \exp[\mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x})]}.$$

From Equation 4, we can see that a latent labeling $\mathbf{h} \in \mathbf{y}^*$ with higher probability can “dominate” the gradient with higher degree. Since the LCRF model is trained with gradient ascent, a latent labeling $\mathbf{h} \in \mathbf{y}^*$ with higher probability will in turn be updated with more gains in the next gradient ascent step (because it “dominated” the current gradient with more degrees). Hence, we expect latent labelings will have a “rich get richer” trend during the training of a LCRF model. This “rich get richer” trend is also meaningful in real-world data and tasks, because in this way the latent structure can be discovered with higher confidence.

On the other hand, note that the probability is expected to be concentrated *highly*, but not *completely*. This is because real-world tasks are usually ambiguous. Another reason is from regularization terms of the objective function, which controls overfitting, including the potential overfitting of latent structures.

B. Latent-Dynamic Inference

Based on the highly (but not completely) concentrated probabilities in LCRFs, we propose a novel inference method, which is efficient and exact in most of the real-world applications.

```

1: Definitions:
2: “ $n$ ” represents the current search step (# of latent-labelings
   being searched).
3: “ProbGap” is a real value recording the difference between
    $P(\mathbf{y}')$  and  $P_{remain}$ .
4: “ $\mathcal{S}$ ” indicates a set of “already searched labelings”.
5: “FindLatentLabeling( $n$ )” uses  $A^*$  search to find the  $n$ ’th
   ranked latent-labeling.
6: “FindParentLabeling( $\mathbf{h}$ )” finds the corresponding label-
   ing from the latent-labeling: FindParentLabeling( $\mathbf{h}$ ) =
    $\mathbf{y} \iff h_j \in \mathcal{H}(y_j) \text{ for } j = 1 \dots m$ .
7:  $P(\mathbf{h}) \triangleq P(\mathbf{h}|\mathbf{x}, \mathbf{w})$ .
8:  $P(\mathbf{y}) \triangleq P(\mathbf{y}|\mathbf{x}, \mathbf{w})$ .
9:
10: Initialization:
11:  $n = 0$ ; ProbGap =  $-1$ ;  $P(\mathbf{y}') = 0$ ;  $\mathcal{S}_0 = \emptyset$ .
12:
13: Procedure LDI():
14: while ProbGap < 0 do
15:    $n = n + 1$ ;
16:    $\mathbf{h}_n = \text{FindLatentLabeling}(n)$ ;
17:    $\mathbf{y}_n = \text{FindParentLabeling}(\mathbf{h}_n)$ ;
18:   if  $\mathbf{y}_n \notin \mathcal{S}_{n-1}$  then
19:      $\mathcal{S}_n = \mathcal{S}_{n-1} \cup \{\mathbf{y}_n\}$ ;
20:      $P(\mathbf{y}_n) = \text{ComputeProbability}(\mathbf{y}_n)$ ;
21:     if  $P(\mathbf{y}_n) > P(\mathbf{y}')$  then
22:        $\mathbf{y}' = \mathbf{y}_n$ ;
23:        $P_{remain} = 1 - \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k)$ ;
24:       ProbGap =  $P(\mathbf{y}') - P_{remain}$ ;
25:   else
26:      $\mathcal{S}_n = \mathcal{S}_{n-1}$ ;
27: return  $\mathbf{y}'$ ;
28:

```

Fig. 6. The algorithm of the LDI inference for LCRFs.

C. Latent-Dynamic Inference (LDI-Naive)

In the inference stage, given a test sequence \mathbf{x} , we want to find the most probable label sequence, \mathbf{y}^* :

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x}, \mathbf{w}). \quad (5)$$

For latent conditional models like LCRFs, the \mathbf{y}^* cannot directly be produced by the Viterbi algorithm, because of the incorporation of latent variables.

In this section, we describe an exact inference algorithm, the latent-dynamic inference (LDI), for producing the optimal label sequence \mathbf{y}^* on LCRFs (see Figure 6). In short, the algorithm generates the best latent-labelings in the order of their probabilities. Then, the algorithm maps each of these latent-labelings to its associated labelings and uses a dynamic programming method (the forward-backward algorithm) to compute the probabilities of the corresponding labelings. The algorithm continues to generate the next best latent-labeling and the associated labeling until there is not enough probability mass left to beat the best labeling.

In detail, an A^* search algorithm [22], [23] with a Viterbi heuristic function is adopted to produce top- n latent-labelings,

$\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n$. In addition, a forward-backward-style algorithm is used to compute the probabilities of their corresponding labelings, $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$. The algorithm then tries to determine the optimal labeling based on the top- n statistics, without enumerating the remaining low-probability labelings, in which the number is exponentially large.

The optimal labeling \mathbf{y}^* will be \mathbf{y}' when the following “exact-condition” is achieved:

$$P(\mathbf{y}'|\mathbf{x}, \mathbf{w}) - \left(1 - \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w})\right) \geq 0, \quad (6)$$

where \mathbf{y}' is the most probable label sequence in the current stage. It is straightforward to prove that $\mathbf{y}^* = \mathbf{y}'$, and further search is unnecessary. This is because the remaining probability mass, $1 - \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w})$, cannot beat the current optimal labeling in this case.

Theorem 4. *In the procedure defined in Figure 6, the labeling \mathbf{y}' (satisfying the exact condition Eq. 6) is guaranteed to be the exact optimal labeling:*

$$\mathbf{y}' = \underset{\mathbf{y}}{\operatorname{argmax}} P(\mathbf{y}|\mathbf{x}, \mathbf{w}).$$

The proof of Theorem 4 is simple. Given the *exact condition*, suppose there is a label sequence \mathbf{y}'' with a larger probability, $P(\mathbf{y}''|\mathbf{x}, \mathbf{w}) > P(\mathbf{y}'|\mathbf{x}, \mathbf{w})$, then it follows that $\mathbf{y}'' \notin \mathcal{S}_n$. Since $P(\mathbf{y}''|\mathbf{x}, \mathbf{w}) > P(\mathbf{y}'|\mathbf{x}, \mathbf{w})$ and $\mathbf{y}'' \notin \mathcal{S}_n$, it follows that $P(\mathbf{y}''|\mathbf{x}, \mathbf{w}) + \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w}) > P(\mathbf{y}'|\mathbf{x}, \mathbf{w}) + \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w})$. According to the exact condition, it follows that $P(\mathbf{y}'|\mathbf{x}, \mathbf{w}) + \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w}) \geq \left(1 - \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w})\right) + \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w})$. The right side of the inequality is 1. Therefore, we have $P(\mathbf{y}''|\mathbf{x}, \mathbf{w}) + \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k|\mathbf{x}, \mathbf{w}) > 1$, which is not possible. Hence, the assumed \mathbf{y}'' is impossible.

D. Admissible and Tight Heuristics for Efficient Search

We have presented the framework of the LDI inference. Here, we describe the details on implementing its crucial component: designing the heuristic function for the A^* heuristic search. Our heuristic search aims at finding the top- n most probable latent-labelings. Recall that the probability of a latent-labelings is defined as

$$P(\mathbf{h}|\mathbf{x}, \mathbf{w}) = \frac{\exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x})\}}{\sum_{\mathbf{h}' \in \mathcal{H} \times \dots \times \mathcal{H}} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{h}', \mathbf{x})\}}.$$

To find out top- n most probable latent-labelings, an easier way for achieving the same target is to find out top- n “highest-score” latent-labelings with the score defined as

$$\varphi(\mathbf{h}|\mathbf{x}, \mathbf{w}) = \mathbf{w}^\top \mathbf{f}(\mathbf{h}, \mathbf{x}).$$

In A^* search, the cost function is normally defined as:

$$f(i) = g(i) + \text{heu}(i),$$

where i is the current node. $f(i)$ is the cost function. $g(i)$ is the cost from the start node to the current node. $\text{heu}(i)$ is the estimated cost from current node to the target node. If the heuristic function is *non-admissible*, the A^* algorithm may

```

1: Procedure LDI-Bounded( $n'$ ):
2:   while ProbGap < 0 do
3:      $n = n + 1$ ;
4:     if  $n > n'$  then
5:       return  $y'$ ;
6:      $h_n = \text{FindLatentLabeling}(n)$ ;
7:     ...
8:   return  $y'$ ;
9:

```

Fig. 7. The algorithm of the LDI-Bounded inference for LCRFs. Since a majority of the steps are similar to the Figure 6, we do not repeat the description. The new input variable n' ($n' \geq 1$) represents the threshold value for bounding the search steps.

overlook the optimal solution [23]. In our case, a heuristic is *admissible* if it never underestimates the scores from the current position to the target. Note that, admissible heuristics do not guarantee the efficiency of the search. In our LDI case, we not only want admissible heuristics, but also try to make the search efficient enough. For this concern, we need a *monotone* (or, *consistent*) heuristic. A monotone heuristic is an admissible heuristic with additional properties. Informally, we can think a monotone heuristic is an *admissible and tight* heuristic. A formal definition of monotone heuristics is as follows for the highest-score path search problem:

$$\begin{aligned} \text{heu}(j) &\geq c(j, k) + \text{heu}(k), \text{ and} \\ \text{heu}(G) &= 0, \end{aligned}$$

where j is every possible current node, and k is every possible successor of j generated by any possible action $a(j, k)$ with the cost $c(j, k)$. G is the goal node. Here we present a monotone heuristic function for the LDI task. The LDI algorithm first scans in a backward direction (right to left) to compute the monotone heuristic function for each latent variables in the lattice. After that, the A^* search was performed in a forward direction (left to right) based on the computed heuristics. For a latent variable h^j on the position i , its monotone heuristic function is designed as follows: $\text{heu}(i, j) = \max_{\mathbf{h}' \in \mathcal{L}(i, j)} \varphi(\mathbf{h}' | \mathbf{x}, \mathbf{w})$, where $\mathcal{L}(i, j)$ represents a set of all possible partial latent-labelings starting from the latent variable $h^j \in \mathcal{H}$ on position i and ending at the goal position m . In implementation, the heuristics are computed efficiently by using a Viterbi-style algorithm:

(1) *Initialization* :

for $j = 1, \dots, |\mathcal{H}|$, $\text{heu}(m, j) = 0$.

(2) *Recursion* (for $i = m - 1, \dots, 1$) :

for $j = 1, \dots, |\mathcal{H}|$,

$$\text{heu}(i, j) = \max_{k=1, \dots, |\mathcal{H}|} [\text{heu}(i+1, k) + \Phi_n(i+1, k) + \Phi_e(i, j, k)].$$

The $\Phi_n(i+1, k)$ is the score of the latent variable h^k on the position $i+1$; the $\Phi_e(i, j, k)$ is the score of the edge between the latent variable h^j on the position i and the following latent variable h^k on the next position.

E. A Bounded Version (LDI-Bounded)

By simply setting a threshold value on the search step, n , we can derive a bounded version of the LDI; i.e., LDI-Bounded (see Figure 7). This method is a straightforward way for approximating the LDI. We have also tried other methods for approximation. Intuitively, one alternative method is to design an approximated “exact condition”, by using a factor, α , to estimate the distribution of the remaining probability:

$$P(\mathbf{y}' | \mathbf{x}, \mathbf{w}) - \alpha \left(1 - \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k | \mathbf{x}, \mathbf{w}) \right) \geq 0.$$

For example, if at most 50% of the unknown probability, $1 - \sum_{\mathbf{y}_k \in \mathcal{S}_n} P(\mathbf{y}_k | \mathbf{x}, \mathbf{w})$, can be distributed on a single labeling, we can set $\alpha = 0.5$ to make a loose condition to stop the inference. At first glance, this seems to be quite intuitive. However, when we compared this alternative method with the LDI-Bounded method, we found that the performance and speed of the former method was worse than for the latter.

F. Existing Inference Methods on Latent Conditional Models

In [13], the optimal labeling is approximated by using a modified Viterbi inference (MVI) method. In the MVI inference, there are two steps. First, the MVI searches for the optimal latent-labeling using the Viterbi algorithm:

$$\mathbf{h}^* = \underset{\mathbf{h}}{\text{argmax}} P(\mathbf{h} | \mathbf{x}, \mathbf{w}).$$

Then, a labeling \mathbf{y} is derived by directly locating the corresponding labeling of the latent-labeling \mathbf{h}^* :

$$\mathbf{y} = \text{FindParentLabeling}(\mathbf{h}^*),$$

which means that $h_j \in \mathcal{H}(y_j)$ for $j = 1 \dots m$. The MVI inference can be seen as a simple adaptation of the traditional Viterbi inference in the case of latent conditional models.

In [4], \mathbf{y}^* is estimated by a point-wise marginal inference (PMI) method. To estimate the label y_j of token x_j , the marginal probabilities $P(h_j | \mathbf{x}, \mathbf{w})$ are computed for all possible latent variables $h_j \in \mathcal{H}$. Then the marginal probabilities are summed up (according to the association between latent variables and labels) for computing $P(y_j | \mathbf{x}, \mathbf{w})$ for all possible labels $y_j \in \mathcal{Y}$. In this way, the optimal labeling is approximated by choosing the labels with the maximum marginal probabilities at each position j independently:

$$\mathbf{y} = \underset{y_j \in \mathcal{Y}}{\text{argmax}} P(y_j | \mathbf{x}, \mathbf{w}) \quad \text{for } j = 1, \dots, m,$$

where

$$P(y_j | \mathbf{x}, \mathbf{w}) = \frac{\sum_{\mathbf{h} \in \mathcal{H}(y_j)} P(\mathbf{h} | \mathbf{x}, \mathbf{w})}{\sum_{\mathbf{h} \in \mathcal{H}} P(\mathbf{h} | \mathbf{x}, \mathbf{w})}.$$

The LDI-Naive and the LDI-Bounded perform exact inference or almost-exact inference, while the MVI and the PMI perform a rough estimation on \mathbf{y}^* . We will compare the different methods via experiments in Section ??.

G. Comparison with MAP Algorithms

The MAP problem refers to finding the Maximum a Posteriori hypothesis, which aims at finding the most likely configuration of a set of variables in a Bayesian network, given some partial evidence about the complement of that set. Several researchers have proposed algorithms for solving the MAP problem [24], [20], [25], [26].

In [20], an efficient approximate local search algorithm is proposed for approximating MAP: *hill climbing* and *taboo search*. Compared to the approximate local search algorithm, the LDI algorithm can perform exact inference under a reasonable number of search steps (with a tractable cost). In the case that exactitude is required, this characteristic of the LDI algorithm is important. In [26], a dynamic weighting A^* (DWA^*) search algorithm is proposed for solving MAP in Bayesian networks. Like the local search algorithms, the DWA^* search is an approximate method and it does not guarantee an exact solution. In [24], an effective method is proposed to compute a relatively tight upper-bound on the probability of a MAP solution. The upper bound is then used to develop a branch-and-bound search algorithm for solving MAP exactly. Whether or not the branch-and-bound search can be used for solving LCRFs is unclear, because of the structural difference. In addition, the quality of tightness of the computed bound is crucial for the tractability of the branch-and-bound search. The quality of tightness is unclear concerning LCRFs.

V. CONCLUSIONS

We made a formal analysis of the inference in latent conditional models, and showed that it is an NP-hard problem, even when latent conditional models have a disjoint assumption and linear-chain structures. More importantly, based on an observation of probability concentration, we proposed the latent-dynamic inference method (LDI-Naive) and its bounded version (LDI-Bounded), which are able to perform exact and fast inference in latent conditional models, even though the original problem is NP-hard.

REFERENCES

- [1] S. Petrov and D. Klein, "Discriminative log-linear grammars with latent variables," in *Proceedings of NIPS'08*. MIT Press, 2008, pp. 1153–1160.
- [2] S. B. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell, "Hidden conditional random fields for gesture recognition," in *Proceedings of CVPR'06*, 2006, pp. 1521–1527.
- [3] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *Proceedings of NIPS'04*, 2004.
- [4] L. Morency, A. Quattoni, and T. Darrell, "Latent-dynamic discriminative models for continuous gesture recognition," in *Proceedings of CVPR'07*, 2007, pp. 1–8.
- [5] X. Sun, L.-P. Morency, D. Okanohara, and J. Tsujii, "Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference," in *Proceedings of COLING'08*, Manchester, UK, 2008, pp. 841–848.
- [6] X. Sun, N. Okazaki, and J. Tsujii, "Robust approach to abbreviating terms: A discriminative latent variable model with global information," in *Proceedings of the ACL'09*, Suntec, Singapore, August 2009, pp. 905–913.
- [7] D. Yu, L. Deng, and A. Acero, "Hidden conditional random field with distribution constraints for phone classification," in *Proceedings of InterSpeech'09*, 2009.
- [8] Y. Wang and G. Mori, "Max-margin hidden conditional random fields for human action recognition," in *Proceedings of CVPR'09*, 2009.
- [9] S. Petrov, "Products of random latent variable grammars," in *Proceedings of NAACL'10*, 2010.
- [10] X. Sun, Y. Zhang, T. Matsuzaki, Y. Tsuruoka, and J. Tsujii, "A discriminative latent variable chinese segmenter with hybrid word/character information," in *Proceedings of NAACL-HLT'09*, 2009, pp. 56–64.
- [11] X. Sun, T. Matsuzaki, D. Okanohara, and J. Tsujii, "Latent variable perceptron algorithm for structured classification," in *Proceedings of IJCAI'09*, 2009, pp. 1236–1242.
- [12] X. Sun and J. Tsujii, "Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation," in *Proceedings of EACL'09*, Athens, Greece, March 2009, pp. 772–780.
- [13] T. Matsuzaki, Y. Miyao, and J. Tsujii, "Probabilistic CFG with latent annotations," in *Proceedings of ACL'05*, June 2005, pp. 75–82.
- [14] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1848–1852, 2007.
- [15] M. R. Garey and D. S. Johnson, *Computers and Intractability*. Freeman, 1979.
- [16] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.
- [17] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation*. Prentice-Hall, 1981.
- [18] G. E. Barton, R. C. Berwick, and E. S. Ristad, *Computational Complexity and Natural Language*. MIT Press, 1987.
- [19] R. B. Lyngsø and C. N. S. Pedersen, "The consensus string problem and the complexity of comparing hidden markov models," *Journal of Computer and System Sciences*, vol. 65, no. 3, pp. 545–569, 2002.
- [20] J. Park and A. Darwiche, "Complexity results and approximation strategies for MAP explanations," *Journal of Artificial Intelligence Research (JAIR)*, vol. 21, pp. 101–133, 2004.
- [21] K. Sima'an, "Computational complexity of probabilistic disambiguation," *Grammars*, vol. 5, no. 2, pp. 125–151, 2002.
- [22] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost path," *IEEE Trans. On System Science and Cybernetics*, vol. SSC-4(2), pp. 100–107, 1968.
- [23] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.
- [24] J. D. Park and A. Darwiche, "Solving MAP exactly using systematic search," in *Proceedings of UAI'03*. Morgan Kaufmann, 2003, pp. 459–468.
- [25] C. Yanover and Y. Weiss, "Finding the m most probable configurations using loopy belief propagation," in *Proceedings of NIPS'03*. MIT Press, 2003.
- [26] X. Sun, M. J. Druzdzel, and C. Yuan, "Dynamic weighting a^* search-based MAP algorithm for bayesian networks," in *Proceedings of IJCAI'07*, 2007, pp. 2385–2390.